MACHINE LEARNING FOR ARCHITECTS AND DESIGNERS: Implementing Machine Learning into the Digital Design Process

Adam Sebestyen
Vienna University of Technology
New Design University St. Pölten

Abstract: In 1962, engineer Douglas Engelbart proposed overlapping the creative mind with artificial intelligence to create designs which could not be created by either entity alone (Engelbart 1962). Today Machine Learning (ML) has entered the public consciousness emerging as an important tool in many industries. Architects should understand these tools to be able to create new and innovative design ideas to meet complex design criteria.

According to Hebron (2016) traditional design algorithms rely on the information programmed into the design software combined with a specific user input/workflow. These systems allow the computer program's behavior to be defined as a finite set of rules that will behave in a predictable manner and thus conform to the programmer's or user's intentions. In comparison ML can detect patterns inside observed workflow data and provide mechanisms for imparting experiential knowledge upon computer systems.

In the specific case of parametric design, rules are established by the user by defining a sequential step-by-step instruction set of geometrical operation tasks upon a set of input data. However, establishing these rules can be a time consuming and complex task. ML can help create those specific rules, if the user can define and provide the necessary input-date and desired output-data. This could lead to faster simulation and optimization methods, as well as the discovery of new parametric design rules.

This paper aims to break down basic ML concepts and proposes how they could be implemented in the architectural digital design process. The focus will be put on supervised machine learning as a tool in aiding and complementing parametric design tasks. A prototype project will be showcased.

The foremost aim of this paper is to lay out the hypotheses of how ML could be further implemented inside the digital design process.

Further, an overview will be given of basic ML and parametric design principles, as well as demonstrating the need for architects and designers to implement ML in their design workflow.

Keywords: Machine learning, artificial intelligence, computational design, digital design workflow

INTRODUCTION

Computer based design has evolved to a point where designers describe geometrical modeling processes, instead of using additive methods inside CAD software. This is done using parametric design, where a modeling process is broken down into individual sequential instruction sets. This method gives the user great flexibility when experimenting with design proposals and/or geometrical shape creation. However, the nature of parametric design requires a lot of idle time when updating or changing complex design processes. This can hamper the creative design process when designers must wait every time they update their design solutions. Further, new design methods or outputs require a new set of parametric design instruction sets. Figure 1 illustrates an example where the parametric design script takes roughly one minute to update.

Machine Learning (ML), as a subfield of Artificial Intelligence (AI), could be implemented inside the digital design process in order to speed up the time needed when calculating complex parametric design workflows and, thus, let the designer work in a more efficient manner. Further, ML could give the designer the ability to explore more design ideas, as well as produce design outputs of higher complexity than would be possible with traditional digital design tools and approaches.

The overall work of my PhD thesis aims to implement ML in the field of digital design practice by developing new workflow methods for the exploration of geometrical outputs. This paper is a starting point for this research process and aims to do the following:

- Give an overview about different ML approaches, recap the basic parametric design principles and outline how implementing ML could help to reduce idle time during geometrical workflow tasks (Section 2).

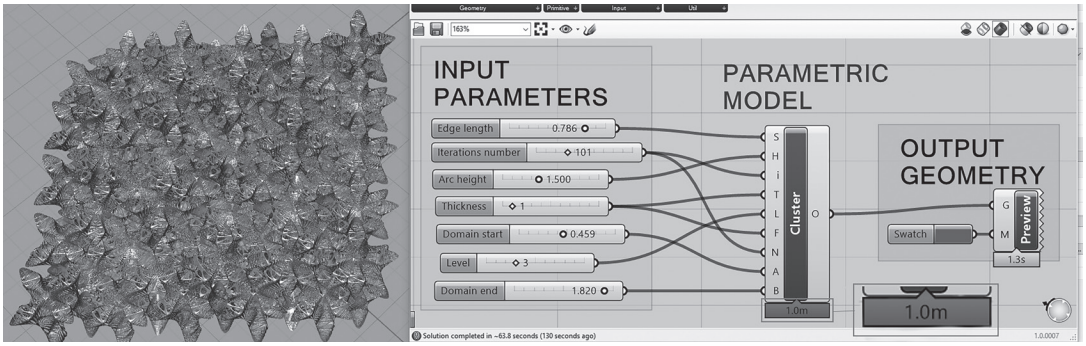

Figure 1: Example of complex parametric design script that requires about one minute to update after input changes. (Author 2019)

- Showcase concepts aimed to speed up the workflow of heavy computational parametric design tasks by implementing ML to let the computer detect patterns in design workflow data. Thus, the ML model would be able to predict the final output that stands at the end of a parametric design script. This would mean a direct conversion from input data to a desired digital design output, without the need to do any intermediary geometrical calculations (Section 3).
- Demonstrate this concept with a parametric facade prototype, where with the help of ML, the relationship of geometrical configuration to certain performance-measure outputs are taught to the computer (Section 3.4)
- Propose two further theoretical concepts of ML based methods to discover unknown parametric design workflows and outputs. One possibility discussed would be merging or blending different workflows. Also, the idea will be discussed of linking geometrical properties to data, in order to discover new and unknown geometrical configurations (Section 4).

1. BACKGROUND AND CONTEXT

In 1962 engineer and inventor Douglas Engelbart proposed the concept of the creative human mind overlapping with artificial intelligence to create designs which could not be created by either entity alone (Engelbart 1962). He envisioned a symbiosis between human and machine; thus, both becoming co-creators in a new and dynamic design process. The goal of such a system would be finding hidden or unknown design languages, methods, or concepts (Wood 2018).

Khean, Fabbri, and Haeusler (2018) argue that although ML has been an essential part in multiple industries and disciplines, in the field of architecture, it has a hard time gaining ground. They state that the field of architecture is "objectively one of the slowest industries to integrate with machine learning." Moreover, they attribute this among other things to the fact that "machine learning expertise can be separate from professionals in other fields; however, this separation can be a major hindrance in architecture, where interaction between the designer and the design facilitates the production of favorable outcomes" (95). Cudzik and Radziszewski (2018) further back this notion, "Despite the growth of machine learning usage, the architectural practice still relies on a daily basis on computer aided drawing and building information modeling" (81).

Hebron (2016) explains how in the field of AI it is especially ML that works on the principle of identifying patterns in observed data, compared to traditionally used design algorithms, which rely on the information being programmed into the design software combined with a specific user input/workflow. These systems allow the program's behavior to be defined as a finite set of rules that will behave in a predictable manner and thus conform to the programmer's or user's intentions. Machine learning, on the other hand, provides mechanisms for imparting experiential knowledge upon computer systems. This lets the computer deal with fuzzier or less precise input data. For example, in the field of 3D geometrical design this suggests that instead of using an explicit set of rules or instructions to describe how geometry should be created or transformed, a machine learning algorithm could look for patterns within a set of sample behavior, in order to produce an approximate representation of the 3D modeling rules themselves. This presents a significant paradigm shift, not only in terms of how future architects and designers will interact with computers and design software but could also fundamentally question long standing principles such as basic design rules and human creativity.

Muklashy (2018) analyzed the potential impact of ML in the architectural field and argues that, although the architectural workload will dramatically change in the future, designers will be "finding more time for creativity while computers handle data-based tasks." He

acknowledges that those designers who are unwilling or unable to adapt "will have trouble pivoting from traditional roles." But for those who see ML as "tools rather than obstacles (it) can lead (them) to freedom from the constraints of old models." He further quotes Mike Mendelson, certified instructor and curriculum designer at the Nvidia Deep Learning Institute,

> Computers are not good at open-ended creative solutions; that's still reserved for humans, … But through automation, we're able to save time doing repetitive tasks, and we can reinvest that time in design.

He also cites Jim Stoddart of the Autodesk design research studio *The Living:*

> We can still leverage the things that humans are really good at—the human intelligence, the creativity - but then also leverage the machine intelligence, the specific capabilities for computers to solve problems really quickly,… a hybrid approach that is actually better than what we're able to do with one or the other separately. (Muklashy 2018)

Therefore, architects should start to understand and implement these tools inside their workflow to be able to create new and innovative design-ideas to meet complex design criteria. Furthermore, research linking the creative practice field and machine learning must be established to find future workflow- and workload-principles, as well as explore the ramifications of ML in the field of design practice.

2. MACHINE LEARNING IMPLEMENTATION IN PARAMETRIC DESIGN WORKFLOWS

2.1. CLASSICAL PARAMETRIC DESIGN APPROACHES

Tedeschi (2014) explains the evolution of computer-based design. He compares early CAD applications to the practice of drawing by hand on paper calling these methods an "additive process", since independent signs of information are overlapped onto each other to convey meaning: "…the drawing is not a smart medium, but rather a code based on standards and conversations" (16).

Advanced CAD applications and parametric design environments let the designer establish relationships between different inputs and workflow steps (figure 2). Tedeschi calls this "algorithmic modeling" describing an algorithm as a "step-by-step procedure" performed by the computer "through a finite list of basic and well-defined instructions". He states that an algorithm "is an unambiguous set of properly defined instructions" and that an algorithm "expects a defined set of input". The benefit over the additive design process is that "algorithmic design enables the user to design a process rather than just a single object" (Tedeschi 2014, 22-25).

Although these design methods allow us to create designs not conceivable with classical additive methods, this process can be very computational heavy and therefore often time intensive. A set of instructions acts upon input data, and in the case of digital design these instructions often entail multiple geometrical operations. This means that on every geometrical or numerical input provided into the algorithmic system, all instructions must be performed individually in sequential order. Therefore, multiple computational geometrical operations will require more time until the final instruction has been completed and the final output can be produced. Further changing just one input parameter requires the algorithm to recalculate all instructions tied to that parameter. In the end, this means that algorithmic processes which are either computational heavy or have a lot of input parameters will require a lot of time from the designer, especially if they wish to produce multiple possible outputs.

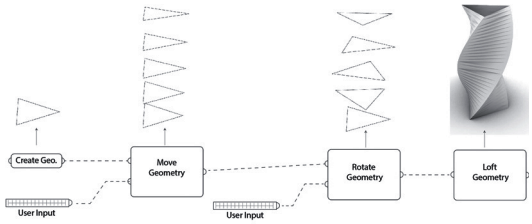

Figure 2: Diagrammatic parametric design process (Author 2019)

2.2. ML BASED PARAMETRIC DESIGN APPROACHES

Using a very broad breakdown, three ML concepts surface, all vastly different in their approach to problem solving and potential design related use cases (Maini and Sabri 2017; Geitgey 2018):

- *Supervised Learning:* These algorithms first must be trained by the user on a known/labeled dataset, teaching the computer what the desired output ought to be for a certain input. After sufficient training, the algorithm will be able to predict the correct outcome corresponding to previously unseen input data.
- *Unsupervised Learning:* Unlike supervised learning, in this case the system does not figure out the correct output but rather helps the user to find the underlying structure, patterns, or meaning in an unlabeled data-set. There is no information available helping the computer to be trained beforehand.
- *Reinforcement Learning:* This algorithm interacts with its environment through an agent. In the absence of existing training data, this agent learns from experience through a trial and error approach in combination with a reward or punishment system. This creates an AI that is eventually able to execute the correct behavior in a wide variety of situations.

I am proposing using Supervised Learning to reduce the time for live parametric model calculations, which will be achieved in two main steps: *training/ building* the ML model and *deploying* the ML model.

In classical parametric design the user establishes rules, i.e., the script that acts upon input data, to produce an (geometrical) output: The user provides the input and rules, the computer generates the output. On the other hand for supervised ML the user would still have to provide the computer with input data, but instead of defining a rule-set to transform the data, the desired output data will be fed into the ML training algorithm. Providing the ML algorithm enough data-sets of input data (X) and corresponding desired output (y) would allow the computer to learn the correct relationship between input and output and thus find the corresponding rule-set. Figure 3 showcases how in classical parametric design the user has to provide input data and a rule-set in order to achieve an output, while in a ML based approach the user would have to define the input and desired output data in order to receive a rule-set. This can be especially useful if the needed rule-set is highly complex.

2.3. ML MODEL TRAINING AND DEPLOYING

The following is a theoretical overview regarding the necessary steps of implementing ML inside the digital parametric design process:

Producing Data for ML Training:

First, training-data must be created: To do so a parametric design script must be built. This script could either produce geometry as its output (parametric modeling process) or analyze geometrical properties (geometrical/performance analysis). In my prototype, which will be discussed later, Grasshopper[1] will be implemented for all parametric design tasks. Next, a multiple of randomized input data would be created and these inputs would have to run through the parametric design script. Afterwards, the inputs and the produced corresponding outputs would be recorded. This step of course could be a time intensive task since many

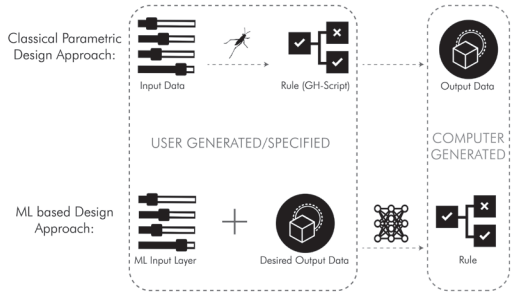

Figure 3: parametric design process (top) compared to ML based design process (bottom). (Author 2019)

calculation circles through a potential heavy instruction set might be necessary.

Training the ML Model:

Next would be the training phase: The supervised ML model would be shown all the created input data together with their corresponding outputs. During the training, the ML algorithm should be able to find patterns in data-shifts from the inputs to the outputs. After training, a successful ML model would be saved.

This step could be time intensive too, depending on the amount of training data used and the specific training settings applied to the ML model. However, the idea is to have the process of generating training-data and building the ML model to be an automated one during which the designer has no active involvement. It would be possible to complete these processes overnight, when the designer is not working, or outsource them to cloud based supercomputers, which do heavy calculations very fast. In essence, the designer is shifting away the heavy time-load they usually must put in while actively changing and experimenting with a parametric design model, to a period when they would not be working.

Deploying the ML Model:

The trained ML Model will be implemented into the parametric design workflow. All components/ instructions used in the original algorithmic script are not needed anymore and can be disabled or deleted. All the user would have to do is connect new input data to the trained ML model. If trained correctly, the model would produce the correct output to the corresponding input data without the need for any time intensive step-by-step calculations. This would give the designer the ability to manipulate or completely replace the input data and receive geometrical or analytical outputs in a very short amount of time. This is possible because, rather than recalculating the entire sequential instruction set, all the ML algorithm must do is to predict corresponding output data according to the current input data.

In essence, ML gives the user the ability of a time tradeoff. Rather than spending time waiting for parametric models to be recalculated and updated during the design process, the user spends time beforehand on building and training an ML model. This step could be done by the computer mostly independently and autonomously. Afterwards during design time, the user will have a parametric tool that will be very fast and efficient to use even on slower computer systems. SideFX (2019) describes a similar approach for the simulations of erosion on digital landscapes: They trained an ML network to generate image data which the 3D FX software Houdini[2] could

interpret as a 3D landscape model. The information regarding the erosion data was part of the produced images. Compared to simulated 3D erosion models the results produced by ML were almost identical but around 50,000 times faster.

3. CONSTRUCTION OF ML BASED PARAMETRIC DESIGN PROTOTYPE: THE NECESSARY STEPS

3.1. DEFINING THE TASK

Preparing data to be usable by an ML algorithm can take a lot of time (Geron 2019). Therefore, thought must be put in beforehand to determine the exact parametric design task and its necessary in and output data.

Tedeschi (2014) describes multiple applications and outputs for parametric design. The most common ones can be summarized as geometrical creations or transformations where, through the parametric design process, geometry is created in a step-by-step process (See section 2.1). The output is almost always a geometrical object. Secondly, there are digital simulations such as cable or membrane simulations. At the output of such a simulation, one often finds a geometrical object as well. A different type of output can be of analytical nature, meaning geometry inside of a parametric design environment, is at some point analyzed and evaluated upon certain criteria, e.g. environmental or structural analysis. Often in parametric form-finding the output of this analysis is used in conjunction with a feedback loop to adapt the geometry and find an optimum based upon the analytical data (a.k.a. optimization).

Thus, we first have to determine the type of ML output (geometrical or analytical) and consider the dimensionality of the input and output data. Geron (2019) explains that ML tasks with multiple outputs are possible, however for basic ML models the user must define beforehand how many output values are expected from a certain amount of input values. For an analytical output this is not a problem, as long as the tool always provides the same amount of output values for different input geometries. However, if the aim is to create geometry as output, it could be difficult to express different geometrical shapes always with the same number of numerical values. A possible solution to this issue would be to represent geometry in a fixed 3D voxel space (Wu et al. 2016). The same applies to the input data: Input data should always be of the same size and dimensionality.

With our proposed prototype we aim to teach the computer the relationship between the geometrical properties of a partial facade system with louvers with its radiation analysis and its light transmittance values.

Overall, there are two main objectives:

- It is hoped that the ML model will correctly learn this relationship, in order to make analytical predictions to a certain geometrical configuration much quicker than could be done through classical simulation.
- Further, it is the hope to utilize this learned relationship to have the ML model predict adequate geometrical configurations according to a user's prior definition of the desired analytical outcome.

For data creation and later ML testing, a Grasshopper script was created. This script mainly does two things: first it produces the facade system geometry based on eight input values; secondly it analyzes the average solar radiation received in kwh/m^2 as well as the percentage of the overall facade system occluded by solid louvers. Thus, in our system we have geometrical properties as input and analytical data as output. Figure 4 shows the parametric design model. The image in the middle shows the model of the facade and its visualized analytical radiation data. The left image shows all eight input values needed to describe the facade geometry; the right shows the corresponding two analytical output values which the script calculated.

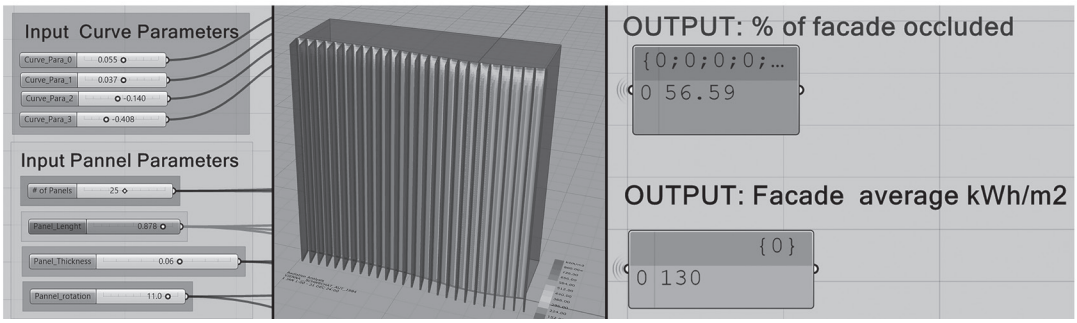

Figure 4: the parametric design script; Left: 8 input values; Middle: the geometrical facade model with visualized analysis; Right: 2 output values. (Author 2019)

3.2. CHOOSING THE RIGHT SUPERVISED MACHINE LEARNING MODEL

Maini and Sabri (2017) explain that there are two tasks of supervised machine learning in regard to the model's output: Regression and Classification. Regression models predict a continuous numerical value as output, while Classification models assign a label to an input out of a finite pool of possible answers.

A parametric design model producing either geometrical or analytical data is capable of an infinite number of possible outcomes, since every unique input usually produces its corresponding unique outcome. Therefore, implementing a *regression* ML model will be necessary.

Ray (2015) describes various regression models, which are used by ML to make predictions based on the input data. He states that the Linear Regression is one of the mostly widely known techniques. However, since a linear regression model can only find the linear relationship between input and output data, it will not be sufficient for a more complex (non-linear) ruleset. Geron (2019) explains how, through the technique of polynomial regression or Decision Trees regression, non-linear relationships between input and output data can be generated.

3.3. CREATING THE DATA-SET

The facade system consists of a variable number of louvers along a one-sided building envelop. The overall number, width, thickness, and rotation of the louvers can be set by the user (all louvers with same length, thickness and rotation). The one-sided building envelop can be described as a NURBS curve of degree three with its start and endpoint being ten meters apart. This curve can be described by a set of four numbers ranging between -1.0 and 1.0. Each off those four values represents the vertical distance from one of the curve's control points to its base line at 0. Setting all four curve parameters at 0 would produce a straight line, setting all four values to 1.0 would produce a curve that is arching upwards, setting all four values to -1.0 would produce a curve that is arching downwards, etc. Expressing the curve in numerical values is necessary since the ML model requires numerical values only to be trained. Overall, there are eight input variables to describe the facade's geometry.

For the outputs, the Plug-In Ladybug[3] was utilized to calculate the radiation analysis of predefined points off the facade. To keep the overall process simple, all these values were summed and averaged in order to have a single value expressing the radiation analysis. Further, a custom Grasshopper script was set up to calculate what percentage of the building envelop will be occluded by the facade's louvers.

Therefore, each facade system is expressed by eight input values and produces two numerical output-values regardless of its overall geometrical shape.

Grasshopper was set up to produce all eight input values randomly, record those values, create the facade geometry, calculate its two corresponding analytical values, also record those two values, and finally save all the inputs and outputs inside a CSV-file. This process was autonomously repeated roughly 1700-times during a period of around 24 hours producing datasets for 1700 different facade systems. Figure 5 shows snapshots from multiple random variations of the facade system.

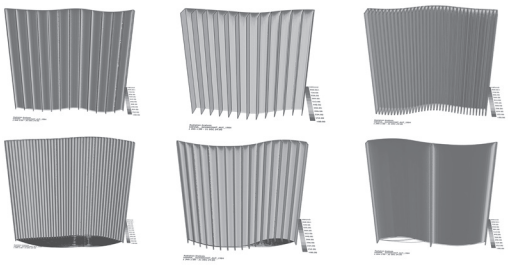

Figure 5: multiple random variations of the facade system used to create ML training and testing data-set. (Author 2019)

3.4. BUILDING THE MACHINE LEARNING MODEL

The Python programming language was used for data perpetration, the ML model's training, testing, and predicting progress. For combining the individual CSV-files and other minor data management tasks, the python library Pandas[4] was implemented. For further data perpetration and the ML training and testing, the Scikit-Learn[5] library was used.

After all the individual data-sets were combined into a single data-set, all inputs and outputs were extracted into their own data-set (*X* for input; *y* for output). Both X and y data-sets where split into training and testing sets (*X_train, X_test, y_train, y_test*) with the training set amounting to about 85% of the overall data. The training set would be used to train the ML model and the testing set would be used to evaluate the performance of the ML model afterwards. Both training and testing input sets were numerically scaled so all their values would fit into the domain from 0 to 1. According to Geron (2019), many ML models require their input data (features) to be scaled and centered.

Two different ML Models were trained, in order to predict different outcomes and perform different tasks:

- For the first ML model, using Scikit-Learn's Random Forest Regression Model we trained with X_train as input data and y_train as output data. Implementing the method of grid-search let Scikit-Learn find good settings for the ML model. The most promising

model produced by the grid-search was saved for later testing. Training a single ML model took around one second. Implementing grid-search took up to two minutes, if the computer was tasked to search among twenty different possible ML setups. Geron (2019) points out that for Random Forest models it is not necessary to provide the algorithm scaled data for training. Therefore, the un-scaled data was used. Nevertheless, it is good practice to produce a scaled data-set as described above in order to be able to test models other than Random Forest (or Decision Tree). However, in our case Random Forest Regression produced the best results. The idea was to implement this model, replacing the slow analytical tools running inside Grasshopper in the hope of quickly producing reliable analytical predictions useable during the early stage design process.

- For the second ML model, training inputs and outputs were *reversed,* meaning that analytical output data (y_train) was fed into the ML model as its *inputs,* while the eight parameters describing the geometrical properties of the facade (X_train) were used as the ML model's training *outputs.* The hope behind this method of training was to produce an ML model that can predict reliable geometrical input data, if the user specifies a desired analytical value *beforehand*. This approach could in theory be applied as an optimization method. Scikit-Learn's Random Forest Regression was used for training the model. As mentioned earlier, it does not require feature scaling. In this case, implementing an ML method not relying on feature scaling should have an advantage when deploying the model, since if a model is trained with scaled data, it also requires scaled data for prediction making. For the first ML model this would not be difficult to do since we as the user set the minimum and maximum values of our parametric design script inputs and thus can easily produce scaled input data. However, since the analytical output data used as input training in the second ML model is not bound by the user, it would be difficult to produce accurate scaled data since we can never know the true minimum and maximum values. A workaround would be to create a sufficiently large data-set with the largest possible wide analytical data spread, and use these values to produce feature scaling.

3.5. TESTING AND EVALUATING THE ML MODEL'S PERFORMANCE

Since 85% of the overall data was used for training both ML models, we were able to use the other 15% for verifying the results:

The first ML model (geometrical property as input, analytical data as output) ended up with a root-mean-squared-error (RMSE) of about 7.1, which is a promising score considering the facade radiation output value ranges from values 23 to 760 and the percentage of facade occlusion from 0 to 100. Using the testing set, for all individual predicted output instances the absolute difference was calculated from the prediction to the true value. For the radiation prediction the average difference was 6.6 points with the biggest difference being 41.1 points. For the occlusion data the average difference from prediction to true value was around 3.1 points with the maximum difference measured being 16.5. Both measurements indicated reliable predictions with the radiation prediction performing somewhat better (figure 7).

The second ML model (analytical data as input and geometrical data as output) achieved a RMSE score of about 2.9. Although it might appear that this model performs more reliably than the first one due to its lower score, this might not be necessarily the case. Most inputs defining the facade geometry exist on a rather small numerical domain. For example, the numerical range of the curve parameters is 2.0, or the numerical range of the louvers' length is 1.0. Further, a geometrical configuration does not necessarily exist for every analytical input specified by the user. Nevertheless, it seems clear that the algorithm was also able to establish a relationship between the analytical outputs and the geometrical input data. Calculating the individual differences from all the test data confirms this notion. The average difference for the curve parameters was around 0.5, which is quite a lot considering the overall range is 2.0 (figure 6). However, the ML model performed much better on the last four parameters describing the louvers' amount, orientation, and dimensions. This might be an indication that the curve parameters are not as decisive for the overall analytical performance of the facade than the other four parameters.

However, it must be pointed out that this method of testing cannot be truly compared to the testing metric of the first ML model. For the first ML model there are always two true analytical values linked to the geometrical facade properties. However, for the second ML model it could be possible that the geometrical configurations predicted by the script might produce similar analytical results as the true geometrical properties, although the predicted and true values do not match. Therefore, those predicted geometrical properties would have to be reinserted inside the Grasshopper script to assess the model's overall reliability. This slow process would need to be repeated multiple times to truly score the model's accuracy

and is outside the scope of this paper (However a few promising verification runs were conducted as described below).

Next, we tested both ML models in conjunction with Grasshopper and new and unseen user provided data:

In multiple runs, the geometrical input data for the facade were set by the user. These inputs were manually fed into the first ML model inside the python programming environment. Simultaneously, the analytical calculations were performed inside Grasshopper. The actual simulation took about one minute inside Grasshopper while the prediction making process using the trained ML model was instantaneous. Just like the RMSE score had suggested, the difference between the actual values and predicted ML values was small and confirmed the differences measured inside the test data. Figure 7 on the left-hand side shows the difference scores calculated using the test data, as well as the prediction and true score inside Grasshopper.

	dif. cur_p1	dif. cur_p2	dif. cur_p3	dif. cur_p4	dif amount	dif length	dif thickness	dif rotation
count	262.000000	262.000000	262.000000	262.000000	262.000000	262.000000	2.620000e+02	262.000000
mean	0.527671	0.534246	0.501084	0.533279	3.823285	0.121208	9.814101e-05	3.926033
std	0.318730	0.318682	0.340551	0.341991	4.178787	0.124596	1.094304e-03	4.592382
min	0.001169	0.001169	0.000659	0.010959	0.000840	0.000105	6.800116e-16	0.003332
25%	0.264428	0.272484	0.214474	0.233553	1.136168	0.033625	8.129504e-08	0.428656
50%	0.512230	0.514465	0.445153	0.497243	2.368519	0.086023	1.514631e-07	1.901939
75%	0.767923	0.776981	0.778427	0.805437	5.128906	0.153243	2.174302e-07	6.460357
max	1.451810	1.451810	1.464245	1.452976	26.643750	0.731154	1.731188e-02	24.800396

Figure 6: testing second ML model: test data information regarding absolute difference between prediction and true values. (Author 2019)

	Dif. in Average_kWh/m2	dif in %_ocluded
count	262.000000	262.000000
mean	6.561556	3.057167
std	6.452155	2.766461
min	0.005556	0.000000
25%	1.881597	0.800490
50%	5.161806	2.244312
75%	9.154514	4.795035
max	41.129167	16.457875

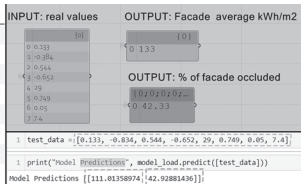

Figure 7: testing first ML model. Left: test data information regarding absolute difference between prediction and true values; Right Top: true Grasshopper calculations; Right Bottom: predictions for the GH data-set above. (Author 2019)

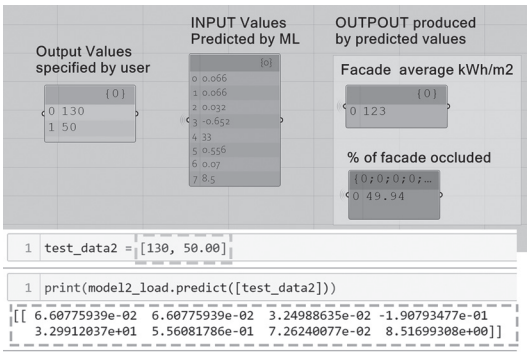

Figure 8: testing second ML model. Top: the user specified analytical values, the geometrical input values predicted by the ML model and the resulting true analytical values; Bottom: The ML input and prediction within Python. (Author 2019)

Since this method achieved reliable information, with a degree of error that seems acceptable during early design phase at a fraction of the time compared to the true simulations, it seems appropriate to do further testing and develop this method more fully in the future.

Assessing the second model is not as trivial as the first. Nevertheless, in the few test runs the predicted geometrical configurations seem to provide reasonable predictions for their targeted analytical performance. In combination with the test data evaluation, this hints to the ML model being able to also recognize the relationship from analytical output data to geometrical input data and justifies a more in depth research of this topic in the future. Figure 8 illustrates an example, where the user was looking for a facade system with the analytical scores of 130 kwh/m^2 and 50%. The predicted geometrical properties ended up producing a facade system with the analytical values of 123 kwh/m^2 and 49.4%.

4. IMPLEMENTING ML FOR EXPLORATION OF NEW DIGITAL DESIGN WORKFLOWS AND GEOMETRICAL SHAPE EXPLORATION

4.1. BLENDING MULTIPLE DIGITAL DESIGN WORKFLOWS IN ORDER TO DISCOVER NEW WORKFLOWS

Maini and Sabri (2017) explain that "linear regression is a parametric method, which means it makes assumptions about the form of the function relating X and Y" (22). Through the implementation of mathematical concepts like a cost function or loss function supervised machine learning can recreate an unknown function by just looking at the function's inputs and outputs. For a function with a one-dimensional input (X) and one-dimensional output (y) using linear regression we would produce a line of best fit. "In three dimensions we would draw a plane, and so on with higher-dimensional hyperplanes" (Maini and Sabri 2017, 22). This is what our ML model described in the previous section would have done if implementing a different ML model than Random Forest Regression (or Decision Tree) like for example a linear regression: In this case, it would have recreated a function by producing a hyperplane with an 8-dimensional input and a 2-dimensional output (and of course for the second ML model a hyperplane with a 2-dimensional input and an 8-dimensional output). Ray (2015) describes more complex regression methods, like the Polynomial Regression, which in 2D space would produce a curve that fits through the data points instead of a line.

Machine learning enables us to express multiple sequential parametric geometrical workflow operations as a single, higher dimensional function. This could be used to potentially discover new forms of geometrical workflows.

The idea is to create two or more geometrical workflows. Each workflow is represented in multidimensional space by its own workflow function. As long as these multiple functions occupy the same multidimensional space, they can be added, subtracted, or blended into each other thus creating new, and eventually unknown geometrical workflows and eventually unpredictable geometrical outputs. Figure 9 demonstrates this in a simplified version in a 2-dimensional space: Workflow Function 1 and Workflow Function 2 exist in the same dimensional space. Therefore, it is mathematically easy to calculate their blended state and thus create a new workflow function.

It is important to note that both workflows do not have to perform the same or similar geometrical operations. They can be vastly different from each other, as long as their input and output dimensional structure are identical.

Figure 9: Blending of two 2D workflow functions into new workflow function (Author 2019)

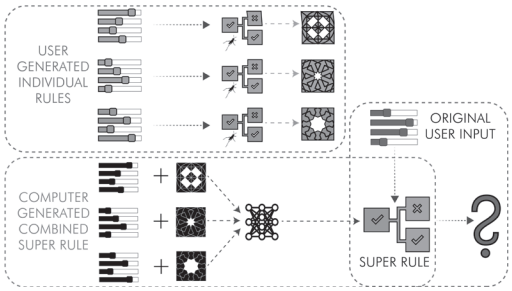

Figure 10: Using ML to combine three individual rule-sets into a single complex super-rule (Author 2019)

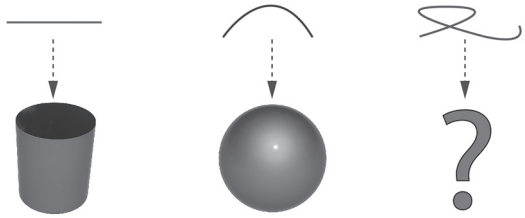

Figure 11: Linking the shape of a curve to 3D geometrical objects (Author 2019)

Figure 10 takes this concept a little bit further: The idea is to combine multiple, individual, and independent rules (parametric design scripts) into a super-rule. For each of the three rules input and output parameters are created just as in our prototype described in the chapter above (again it is important that all three rules have the same amount and dimensionality of input and output data). Instead of training data-sets derived from a single rule individually, all data-sets would be fed into the *same* ML model. This would create the super rule a complex combination of all individual rule-sets.

4.2. RELATING GEOMETRICAL PROPERTIES TO DATASETS IN ORDER TO DISCOVER NEW GEOMETRICAL CONFIGURATIONS

The paragraph above describes a theoretical method of combining multiple workflows in order the create new ones. A somewhat similar approach could be used within a single workflow function to discover new geometrical properties. In classical parametric design the input geometry is inevitably linked to the output geometry. We could also call the output a geometrical evolution of the input–one being the first link in a chain and the other the last link in the same chain.

Machine learning lets us link data that are usually not related. For example, we could easily link the pixel information in an image to the amount of curvature of a surface, or the current temperatures measured in all districts of the city of Vienna to the voxelization density of a mesh geometry. Surely relating data can also be done by traditional programming or parametric design; however, these rules of relationship would first have to be defined/programmed by the user. As Maini and Sabri (2017) explain with machine learning we can ". . . build models [to] predict [outputs] without having explicit pre-programmed rules and models" (9).

This means that the user would have to provide the ML model with a single input data-set and relate it to a desired state of a geometry. Afterwards they would repeat the same step with an altered input data-set and also an altered geometrical state. This can be repeated a couple of times and afterwards the model would be trained. Now the user can feed new input data into the trained ML model, which logical output would be very difficult for a human to predict. However, for the algorithm it will be very easy to produce the unexpected, however correctly corresponding geometrical result. This could be a useful tool in the creation of unknown complex geometrical configurations.

Figure 11 demonstrates this on a simplified version: If a red straight curve would be linked to the geometrical shape approximating a cylinder and a bend blue curve a sphere, what shape would be linked to the looping green curve?

This method has two clear advantages compared to classical parametric modeling:

- As described in section 2.2 training an ML model to perform geometrical operations could potentially save the designer a lot of idle time, especially with complex operations.
- It would be easy to link datasets to geometrical outcome since no explicit rule-set has to be defined by the user but rather desired geometry output states. Further, it would also be easy to link input data to geometrical parameters of different dimensionalities.

CONCLUSION

In this paper, I discussed the potential for implementing supervised machine learning performing regression to create new workflows of digital design. The output could be the creation and/or modification of complex geometries or sped up analysis or optimization tools. Implementing established parametric design approaches to create training data for ML models could be a potential time saver for the designer while they are in the process of exploring geometrical transformations. This could lead to more and complex geometrical experimentations in the field of design studies and creation.

An early prototype was created, training one ML model to predict analytical data based upon input geometry of a facade system and training a second ML model to predict reasonable geometrical configurations capable of producing similar analytical results to those specified by the user beforehand. Both prototypes produced promising results, so future exploration into the topic seems viable.

Nevertheless, the argument could be made that the relationships between the prototype's eight input data values and two output performance scores is not very complex and the workload of producing the data-sets does not justify the end results. Therefore, future focus should be put on more complex parametric relationships with a significant higher amount of input and output values; this could also include entire geometrical configurations as ML outputs. For the current ML model training, a Random Forest model was used. This was fast and sufficient, however for more complex future tasks it seems reasonable that a more complex ML model setup is necessary, such as a deep neural network. ML library *Kears*[6] could potentially provide the ability to create such ML models specifically tailored towards complex geometrical workflow operations.

Besides improving and building upon the existing prototype, time should eventually be dedicated towards exploring the theoretical possibilities discussed in section 4.1 and 4.2: This would be the exploration of blending multiple geometrical workflows into each other, thus creating new and unexpected workflows. In addition research will be dedicated to the topic of linking multidimensional data to geometrical object states, in order to discover unknown states.

ACKNOWLEDGMENTS:

Credit must go to my department colleague *mgr. inz. arch. Jakub Tyc*, who produced all the Grasshopper scripts necessary for the facade performance analyses. He further supported me during the creation of the facade system and the data recording process.

Credits to the icon artists: *akash k; Trevor Dsouza; priyanka; Chameleon Design; Lluisa Iborra.*

ENDNOTES

1 www.grasshopper3d.com
2 https://www.sidefx.com/products/houdini/
3 https://www.ladybug.tools
4 https://pandas.pydata.org
5 https://scikit-learn.org/
6 https://keras.io

REFERENCES

Cudzik, J., Radziszewik, K. 2018. *Artificial Intelligence Aided Architectural Design*, *Proceedings of the 36th eCAADe Conference* - Volume 1, Lodz University of Technology, Lodz, Poland, 19-21 September 2018, 77-84 Available at: http://papers.cumincad.org/cgi-bin/works/Show?ecaade2018_139

Engelbart, D. C. 1962. *Augmenting Human Intellect: A Conceptual Framework.* Menlo Park: Stanford Research Institute. Available at: https://doi.org/10.21236/ad0289565.

Gatys, L., Ecker A., and M. Bethge. 2015. "A Neural Algorithm of Artistic Style." *Journal of Vision* 16, no. 12: 1-16. https://doi.org/10.1167/16.12.326.

Geitgey, A. 2014. "Machine Learning Is Fun! Medium." Accessed November 2019. https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471.

Geron, A., 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. OReilly Media.

Hebron, P. 2016. *Machine Learning for Designers.* Sebastopol, CA: O'Reilly Media. https://www.oreilly.com/library/view/machine-learning-for/9781491971444/.

Khean, N. Fabbri, A. and Haeusler, M. 2018. *Learning Machine Learning as an Architect, How to? Presenting and evaluating a Grasshopper based platform to teach architecture students machine learning*. in Kępczyńska-Walczak A., Białkowski S. (ed.), computing a better tomorrow - 36th eCAADE, Lodz, Poland, pp. 95 - 102, presented at 36th eCAADe conference, Lodz, Poland, 19 September 2018 - 21 September 2018

Maini, V., and Sabri, S. 2017. *Machine Learning for Humans*. https://www.dropbox.com/s/e38nil1dnl7481q/machine_learning.pdf?dl=0.

Miller, N. 2018. "New Machine Learning Examples with LunchBoxML." *Providing Ground.* Accessed November 2019. https://provingground.io/2018/03/12/new-machine-learning-examples-with-lunchboxml/.

Muklashy, W. 2018. "How Machine Learning in Architecture Is Liberating Designers." *Redshift by Autodesk.* Accessed November 2019. https://www.autodesk.com/redshift/machine-learning-in-architecture/.

Ray, S. 2015. "7 Regression Types and Techniques in Data Science." *Analytics Vidhya.* Accessed November 2019. https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/.

SideFX. 2019. "Machine Learning Data Preparation." *SideFX*. Accessed November 2019. https://www.sidefx.com/tutorials/machine-learning-data-preparation/.

Tedeschi, A. 2014. *AAD_Algorithms-Aided Design: Parametric Strategies Using Grasshopper*. Brienza: Le Penseur Publisher.

Wood, H. 2018. "The Architecture of AI." *Medium*. Accessed November 2019. https://medium.com/@hw_/the-architecture-of-ai-a872c520f19c.

Wu, J, et, al. 2016. "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling." Accessed November 2019. https://arxiv.org/abs/1610.07584